Esquematización y Diseño del Flujo ETL

1. Introducción

En este paso, vamos a planificar y diseñar el flujo ETL (Extract, Transform, Load), una técnica fundamental en la gestión de datos que consiste en extraer datos de diversas fuentes, transformarlos según necesidades de negocio o de análisis, y finalmente cargarlos en una base de datos o sistema de destino. El objetivo de este diseño es garantizar que el flujo de datos sea eficiente, escalable y fácil de mantener.

1. Herramientas Elegidas
   1. Lenguaje de Programación: Python

Elección: Python es ampliamente reconocido por su facilidad de uso y flexibilidad. Es una opción ideal para implementar un flujo ETL por varias razones:

* Bibliotecas poderosas: Python ofrece una variedad de bibliotecas como pandas para manipulación de datos, requests para obtener datos de APIs y sqlite3 para interactuar con bases de datos SQLite.
* Comunidad activa: La amplia comunidad de Python ofrece soporte constante, así como una gran cantidad de recursos y tutoriales.
* Escalabilidad: Python es adecuado para trabajar con proyectos pequeños y grandes por igual, lo que garantiza que el flujo ETL sea extensible si el volumen de datos crece en el futuro.
  1. Herramienta de Visualización: Power BI

Elección: Power BI es una herramienta robusta para la visualización de datos y creación de informes interactivos. Es ideal para la fase de presentación de los datos procesados en el flujo ETL:

* Conectividad con SQLite: Power BI puede conectarse de manera directa con SQLite3, lo que facilita la importación de los datos transformados y cargados en la base de datos.
* Interactividad y facilidad de uso: Power BI permite crear dashboards y reportes interactivos sin necesidad de escribir código complejo, lo que lo convierte en una opción accesible para usuarios sin experiencia en programación.
* Optimización de informes: La herramienta tiene capacidades de optimización y análisis rápido de grandes volúmenes de datos.
  1. Base de Datos: SQLite3

Elección: SQLite3 es una base de datos ligera y fácil de usar que ofrece una serie de ventajas para proyectos que no requieren una infraestructura compleja:

* Simplicidad: Es fácil de configurar y no requiere un servidor dedicado, lo que permite un manejo sencillo de datos locales en un entorno de desarrollo.
* Portabilidad: Los archivos de base de datos SQLite son autónomos y pueden moverse entre sistemas fácilmente.
* Rendimiento adecuado para proyectos pequeños a medianos: Para proyectos con volúmenes de datos no excesivos, SQLite3 es una base de datos completamente funcional, permitiendo crear tablas y consultas rápidamente.

1. Flujo ETL Propuesto
   1. Extracción (Extract)

En esta fase, los datos se extraerán desde diversas fuentes, que podrían incluir archivos CSV, APIs o bases de datos externas. En el contexto del proyecto:

* Fuentes de datos: Los datos de tweets (o cualquier otra fuente de datos de interés) serán obtenidos a través de la API de Twitter o archivos locales (como CSV).
* Métodos de extracción: Utilizaremos bibliotecas como requests para obtener datos desde APIs, y pandas para manejar archivos CSV o Excel.
  1. Transformación (Transform)

Una vez que los datos se han extraído, la fase de transformación consiste en:

* Limpieza de los datos: Eliminación de valores nulos, duplicados, y transformación de texto (por ejemplo, convertirlo a minúsculas o eliminar stopwords).
* Formateo: Asegurar que los datos estén en el formato adecuado para análisis o visualización.
* Agregación: Dependiendo de la necesidad, se podrían agregar o combinar datos de diferentes fuentes (por ejemplo, calculando la longitud de los tweets o extrayendo palabras clave).
* Análisis de Sentimiento: Si se usa un modelo de análisis de sentimiento, se realizarán las transformaciones necesarias para agregar una columna de sentimiento a los datos.
  1. Carga (Load)

En la fase de carga, los datos transformados se insertan en una base de datos:

* Base de datos SQLite3: Creamos tablas con las columnas necesarias para almacenar la información que se ha extraído y transformado.
* Inserción de datos: Usaremos la biblioteca sqlite3 de Python para insertar los datos en las tablas de la base de datos.
  1. Visualización

Una vez que los datos están en la base de datos, se conectarán a Power BI para su visualización:

* Conexión a SQLite3: Power BI tiene soporte nativo para conectarse a bases de datos SQLite, permitiendo importar fácilmente los datos almacenados.
* Creación de Dashboards: Los datos se presentarán en dashboards interactivos donde se podrán analizar tendencias, realizar comparaciones o generar informes visuales.

1. Consideraciones Técnicas

* Frecuencia de Ejecución: Dependiendo de la fuente de datos y la necesidad de actualización, el flujo ETL podría ejecutarse de forma diaria, semanal o incluso en tiempo real si los datos lo requieren.

Extracción de Datos desde una API Pública

En esta fase, el objetivo es obtener datos relevantes desde una API pública, lo cual es esencial para iniciar el proceso de análisis. Se seleccionó una API pública específica en función de la naturaleza del proyecto, como las siguientes opciones:

* OpenWeatherMap: Proporciona datos meteorológicos en tiempo real, lo cual es útil si se necesita información sobre el clima actual.
* NewsAPI: Ofrece acceso a noticias en tiempo real de diversas fuentes, ideal para proyectos relacionados con el análisis de tendencias informativas.
* CoinGecko: Proporciona datos sobre criptomonedas, perfecto para proyectos relacionados con el análisis de mercado de divisas digitales.
* Twitter API: Permite acceder a tweets y datos de usuarios de Twitter, muy adecuado para proyectos de análisis de redes sociales.

Para garantizar que la API funcione correctamente, se recomienda utilizar herramientas como Postman para probar las respuestas y asegurar que los datos sean accesibles antes de implementar el código en Python.

Proceso en Python: Se utiliza la librería requests para realizar una consulta HTTP hacia la API seleccionada y obtener los datos en formato JSON. La respuesta de la API es almacenada en variables para su posterior procesamiento.

Transformación de Datos

La transformación de datos es un paso clave en el proceso ETL. En esta fase, los datos extraídos de la fuente se limpian y se ajustan a un formato adecuado para su posterior análisis. Algunas de las tareas comunes en esta etapa son:

* Eliminación de valores nulos: Si los datos contienen valores nulos o vacíos, se tomará una decisión sobre cómo manejarlos. Por ejemplo, se pueden reemplazar por un valor predeterminado o eliminarlos completamente si son poco significativos para el análisis.
* Normalización de formatos: Los datos extraídos pueden tener diferentes formatos (como fechas en distintos estilos o números con comas y puntos), por lo que se estandarizarán para que tengan una estructura uniforme. Esto es crucial para evitar errores en el análisis posterior.
* Corrección de errores evidentes: En algunos casos, los datos pueden contener errores evidentes, como caracteres extraños, duplicados o datos mal introducidos. Estos serán corregidos o eliminados para asegurar la calidad de los datos.
* Conversión de tipos de datos: Es posible que algunos valores necesiten ser convertidos a un tipo de dato específico, como convertir cadenas de texto que representan fechas en objetos de fecha o números que tienen símbolos como comas en valores numéricos.

Estas transformaciones asegurarán que los datos sean consistentes y estén listos para su carga en la base de datos.

4. Carga de Datos

Una vez que los datos han sido limpiados y transformados, el siguiente paso es cargarlos en una base de datos para su almacenamiento y análisis. En este caso, se utilizará SQLite3, una base de datos ligera y fácil de usar para pequeños proyectos. Los pasos en esta fase son:

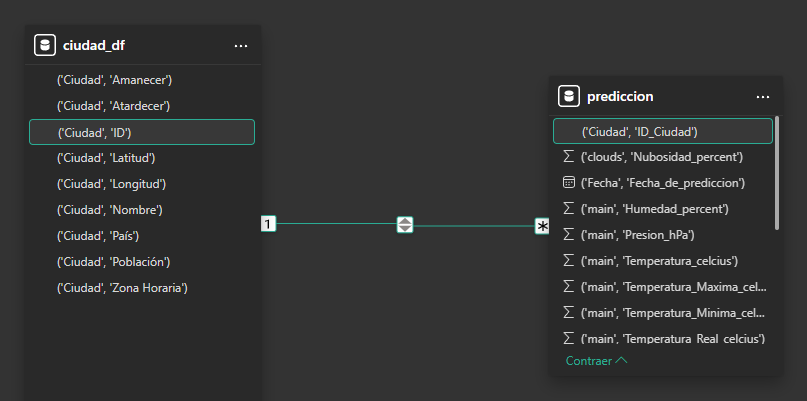
* Creación de la estructura de la base de datos: Primero se crearán las tablas necesarias en SQLite3, asegurando que la estructura de la base de datos esté optimizada para almacenar los datos de manera eficiente. Esto incluye la definición de los tipos de datos y las relaciones entre las tablas.
* Inserción de los datos: Los datos transformados se insertarán en las tablas correspondientes de la base de datos utilizando comandos de inserción SQL. Este proceso puede incluir la verificación de duplicados antes de insertar los nuevos registros para evitar la redundancia de datos.
* Verificación de la carga: Una vez insertados los datos, se realizarán consultas de verificación para asegurar que todos los registros se hayan cargado correctamente y que no haya inconsistencias en la base de datos. Esto puede incluir la ejecución de consultas SELECT para revisar el contenido de las tablas y comprobar que los datos se insertaron correctamente.
* Optimización de la base de datos: Después de la carga, se puede realizar una optimización de la base de datos (como la creación de índices) para mejorar el rendimiento en futuras consultas y análisis.

Modelo Entidad-Relación (ER)

El Modelo Entidad-Relación (ER) es fundamental en el diseño de bases de datos, permite representar de manera visual las entidades involucradas en un sistema y las relaciones entre ellas. Este modelo facilita la organización y estructuración de los datos, asegurando una gestión eficiente de la información. En un diagrama ER, las entidades representan objetos o conceptos clave, mientras que las relaciones describen cómo interactúan estas entidades entre sí.

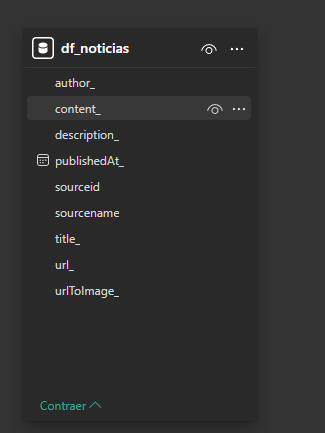
# OpenWeatherMap

En este caso, estamos manejando datos de predicciones meteorológicas en una base de datos y necesitamos asociar las predicciones con ciudades específicas, podemos modificar la estructura de la tabla de predicciones para incluir un campo adicional que almacene el ID\_Ciudad. Esto permitirá asociar cada predicción con una ciudad concreta, lo que es útil en escenarios donde los datos pueden incluir predicciones para múltiples ciudades.



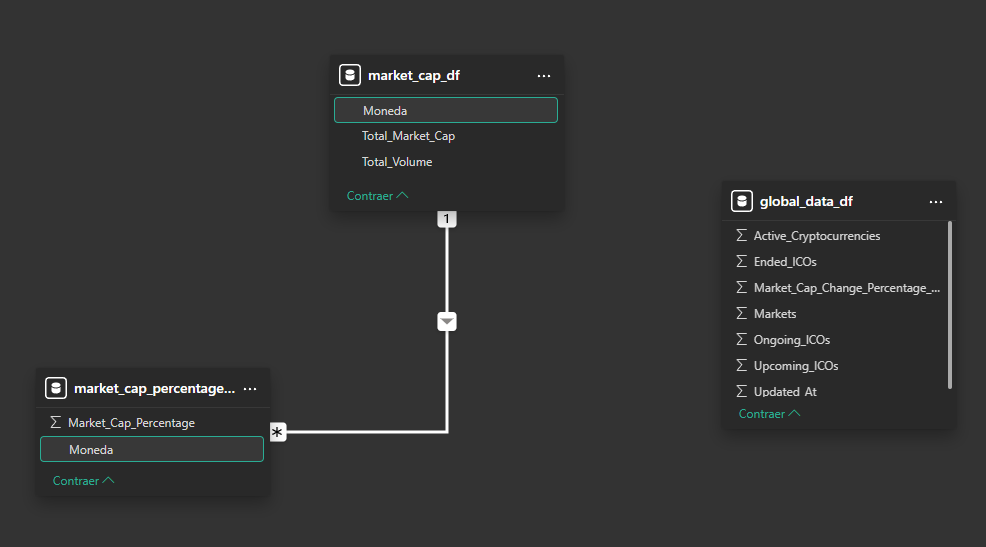
# NewsAPI

En lugar de utilizar múltiples tablas al integrar la API de NewsAPI, he optado por mantener la estructura de nuestra base de datos. Esto se logra con una única tabla que contendrá toda la información relevante sobre las criptomonedas, como la capitalización, el crecimiento, y otros aspectos importantes como el nombre de la moneda.



# CoinGecko

He optado por no utilizar un ID único como clave primaria para las tablas que almacenan información sobre la capitalización y el crecimiento de las monedas, debido al volumen de datos. En lugar de esto, decidí usar el nombre de la moneda como identificador principal. Este enfoque simplifica la estructura y reduce la complejidad de la base de datos.



# Twitter API (Acceso Básico)

En este caso, hemos decidido integrar los datos de la Twitter API de forma simplificada, añadiendo una nueva columna en nuestra tabla principal en lugar de crear una tabla separada. La razón principal es que el volumen de datos relacionado con los tweets es limitado (solo se procesan hasta 100 tweets por vez), lo que hace innecesario crear una tabla adicional y establecer relaciones con claves foráneas como un id\_user.

Si en el futuro los datos comienzan a ser más extensos, podría ser útil considerar la creación de una tabla separada para los tweets individualmente, en este caso pueden ser reposteados por más de uno. Esto permitiría almacenar más detalles, como el autor original, la fecha de publicación y otros atributos, y establecer una relación mediante un id\_user (de tener acceso a los mismos) . Sin embargo, en este momento, con solo 100 tweets, mantener una única columna es suficiente y práctico.

